International Journal of Engineering &Technology, 7 (1.6) (2018) 23-26

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

Optimal component architecture using particle swarm optimization algorithm for self-adaptive software architecture

Y.MohanaRoopa[1]*, M. Ramesh Babu[2], Jetti Kumar [3], D. Kishore Babu[1]

[1]Departement of Computer science and Engineering, Institute of Aeronautical Engineering, India
[2]Departement of Electronics and Communication Engineering, Institute of Aeronautical Engineering
[3]Departmrnt of CSE, Koneru Lakshmaiah Education Foundation, India
**Corresponding author E-mail:ymohanaroopa@gmail.com*

Abstract

The component-based software engineering (CBSE) ensue the procedure of reconfiguration and reusability of components to reap the higher productivity. The context-aware structures are portion of CBSE, which observes the functionality of the system and adopt automatically according to the execution context. In this paper, we are focusing on the aware context guidelines that automatically adapt to the given context given by the customers and remodel the software architecture based totally on the requirements. The component repository turned into added, in which it carries the wide variety of reusable components. The fuzzy logic becomes carried out to the component evaluation in the component repository. The Particle Swarm Optimization (PSO) algorithm applied, to optimize component architecture. The Hospital management system is used to test the adaptability of the system.

Keywords:*Components; PSO; GRASP; Fuzzy Adapter; Context-Aware Systems; Self-Adaptive Software Architecture.*

1. Introduction

The adaptive software architecture works on the component based development. The components are the building parts by combining the software coding [1], [2]. Software adaptation is the process of changes the component functionality according to the user requirements dynamically. It includes the detection of interaction failure and their repair when it is required [3], [4], [5]. Adaptation considered in various aspects like component perspective or service perspective. If it is from the component side, the actors are interfaces, services, and ports, and if it is from service point the actors are interfaces and services.

As per the researchers, the developers of software may combine software components from any business vendors (COTS) [6]. Hence, there is no limit for combining the components from various models. It means the developers have the flexibility to select the components from business vendors. Except that there's a quantity of distinctive service models and components, the application architect needs to select a language that is suitable for component development for appropriate adaptation [7]. Additionally, component models that are not limited to easy call and response; the component includes the component interface to furnish the interactive response and help adaptation facilities which raises the component reusability [23].

In this paper, we address the component evaluation for increasing the reusability. The internal component interaction provided by the component repository. The component evaluation and adaptation is investigated by the fuzzy logic application.

2. Related work

In [8], the authors introduced a KPN models to design the model-controller- adapter framework. This framework is utilized for self-adaptively of the streaming purposes. The work didn't talk about the main points in regards to the monitor and adaptation controllers. In spite of the fact that a contextual investigation is given, no effects are feasible to assess the scheme regarding its adequacy and execution (convergence, design and so forth.).

The common mechanism to self-adaptation furnished in [9], which separation of the concerns systemfunctionalities and self-administration. Self-adaptation is finished by using applying an arrangement of adaptation techniques on programming components even as specified configurable framework triggers these tactics. Conceivable adaptation for part conduct and software parameters is additionally examined.

In [10], authors discuss the examination outcome of the MADAM that has conveyed radical answer to the development, operation of self-adaptive applications and context-aware services. The main commitments of this work are (a) a tricky middleware that backings the runtime adaptation of element-situated items (b) a creative model-driven improvement technique contemplating summary adaptation models and specifications on model-to-program changes.

A self-adaptable framework is proposed in [11], where a proposition to supervise self-adaptation at hardwareand programming phases with the procedure for a decentralized given control algorithms. An objective of exact interface, alongside a decentralized and facilitated control process is proposed as an important thing related to this procedure [22].

In [12], [13], and [14], different ways for quality-grained QoS control of multimedia purposes are exhibited. The recommended procedures create a composed utility that meets adictedQoS prerequisites from an information utility programming. The monitoring mechanism monitors the development of the algorithm in a cycle and picks the following recreation to run and its first stage, guided via security and optimality requirements to the framework.

3. Architecture for component evaluation

Figure 1 explains the architecture for component Evaluation. The evaluator is used for evaluating the given element. The evaluator works with the principle of fuzzy logic and it assessments the adaptability of the given component with the context [15]. The component adaptability ranges from 0 % to 100%. Let us recall that, there is a request for evaluation of component, the evaluator module sends the request for gathering data from the component storage, and the data includes the set of rules. The format of the rules is given below.

If (Condition)
"Component is acceptable."
Else
"Component is not acceptable."

By way of utilizing above rule, still simplify that it's higher to use first order logic for determining the component is acceptable or not. For the generated rules, apply fuzzy logic to identify that the component is average, much or less adequate which used to be good explained in [16]. The evaluator module includes the component storage that stores the component description. The very fact database contains the context knowledge that was once dispatched by way of the client interface for execution.

The fuzzy adapter is introduced in evaluator module that involves the following parameters. The fuzzy set involves the suitable function that the input function may belong.

a) The fuzzy logic involves the mathematical operations
b) The fuzzy rules are associated by the component storage.
c) The Fuzzy set incorporate the inference engine's output variables.

The context environment includes the ontology that is used to characterize the huge principles to the area and the range of these principles (the interface monitors the standards). The fuzzy adapter manages the acceptable stage of the components.

The component descriptor contains parameters of the component which might be associated with component storage. The developer of the component can recognize the evaluator module by using analyzing the component description [16], [17]. The certain clarification Fuzzy adapter is shown in figure 1.The framework in figure 1 examines the software self-adaptation on the context environment. The fuzzy adapter is contact with application logic through designer of architecture. The appliance logic contains the reusable components which might be associated with context environment. In step with the functional requirements, the designer of architecture selected the components from the application logic and evaluated by way of utilizing the fuzzy adapter, the fuzzy adapter involves adaptation goals such as of rules, fuzzification, defuzzification and fuzzy inference.

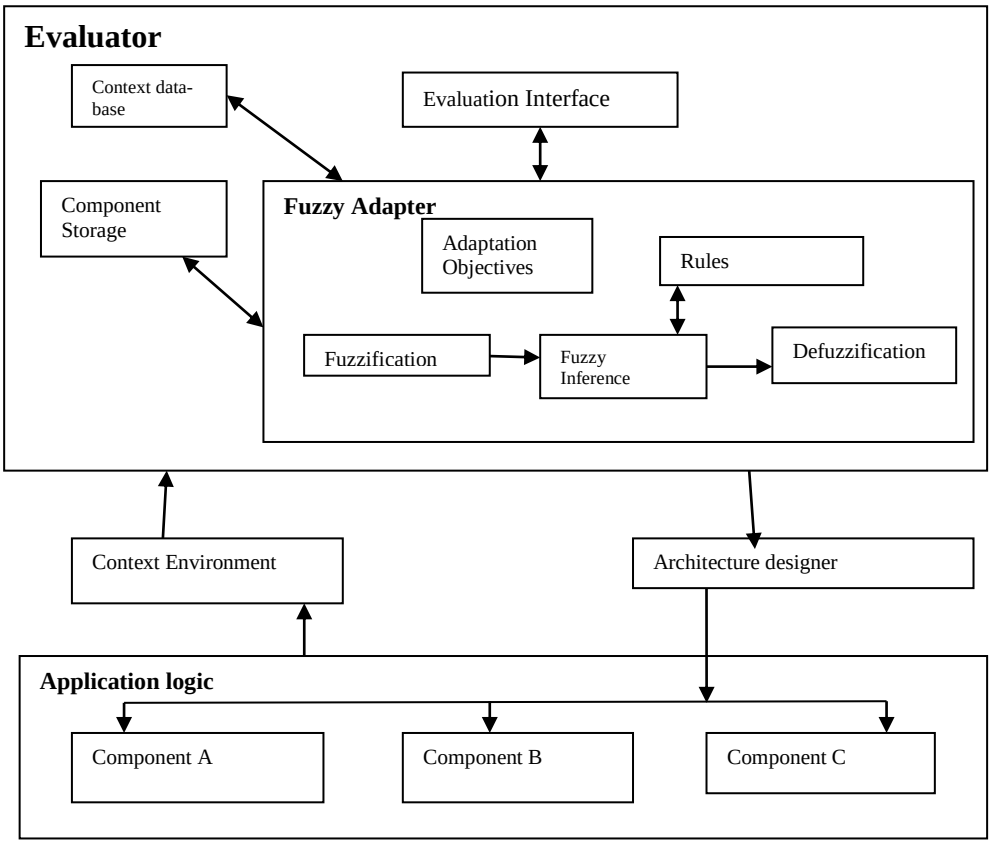

Fig. 1: Framework for Component Evaluation Fuzzy Adaptation Mechanism.

The above evaluation of component system causes some issues even as evaluating the component from various manufacturers. For example, a manufacturer has special that for evaluation of component is 60 on the proposal and a different organization had specified it as 70 for the identical context. In keeping with solve this drawback the optimal component architecture is defined to fit into the software architecture.

4. Optimal component architecture using PSO

Particle Swarm Optimization (PSO) is a computational strategy which optimizes a quandary through iteratively trying to toughen a candidate solution with reference to a given measure of best PSO originated from the simulation of social behavior of birds in a

flock. In PSO, each particle flies within the search house with a speed adjusted with the aid of its own flying reminiscence and its companion's flying expertise. Every particle has its purpose function worth which is determined through a health operate. PSO is an evolutionary computation system which is similar to the Genetic Algorithm. Right here a distinct system is initialized by means of a group of random options.

In Particle Swarm Optimization together with every possible solution, inconstant speed is likewise assigned which represent a particle. Every particle follows its coordinates inside the hassle space in reference to the excellent solution. Here the fitness value is also taken into consideration for the similarly manner. This fitness value is called *pbest*. The vicinity of these answers is regarded as *gbest*. In our proposed method we've got applied a changed version of PSO. In this PSO we've got assigned worst case as well along with the first-class case and also pass over operation is also protected after the fitness choice which could similarly increase the opportunity of choosing the first-rate particle.

The Particle Swarm Optimization gives improved solution and the steps used are presented in the following section.

PSO approach steps:

The different steps used to implement the Particle Swarm Optimization approach,

i) Initialize a particle's group (solutions) with velocity and position selected inconstantly for n-variable in the domain of the problem.
ii) Any of these inconstantly generated particles calculate the fitness functions with n- variables.
iii) Correlate this fitness value with the *pbest* value. If this present fitness value improved than the *pbest* then select the present fitness value as the *pbest* for the further processing.
iv) These fitness values is compared with the overall best previous values and if the present value is improved than update the *gbest* for the current particles array index and value as the new *gbest*.
v) Revise the particle's position and velocity and rerun the steps as far as the criterion of improved fitness value attained. The particle's position and velocity are differed with the help of the below equations,

$$v_j(n+1) = v_j(n) + a_1 d_1(g_j(n) - h_j(n)) + a_2 d_2(g_j'(n) - h_j(n)) \quad (1)$$

$$h_j(n+1) = h_j(n) + v_j(n+1) \quad (2)$$

vi) The procedure reciprocated as far as the improved fitness value solution is attained.

The above said equations contain the constant values of acceleration this is wanted for mixing every particle with the *pbest* and *gbest*. Updating the best position of the particle can be given as per the equation below,

$$g_j(n+1) = \begin{cases} g_j(n), & k(h_j(n+1)) \geq k(g_j(n)) \\ h_j(n+1), & k(h_j(n+1)) < k(g_j(n)) \end{cases} \quad (3)$$

The particle velocity at each dimensions are bounded to the interval are then measured and is compared. This equation is an important parameter, in determining the resolution with which the region between present position and the target position are searched. The above sets of equations are used to calculate fitness of the solution and to choose the good result depends on these fitness values.

So, Particle swarm Optimization used for dealing the optimization problems as different evolutionary algorithms. It could be implemented in extraordinary areas as robotics, simulations applications and signal processing. Here hired PSO so as to choose the best solution for the element optimization. The distinctive components are processed in the algorithm to gain the perfect consequences suitable to software architecture.

Greedy Randomized Adaptive Search Procedure (GRASP) [18] is suitable to get the best solution for NP-complete problems. For optimization of component architecture, we're using the GRASP approach for making improvements to the component accessibility and reliability related to real system.

The GRASP contains the iterative algorithm that is composed of two steps. One is build, and a different one is search. After successive iterations, the algorithms in finding the best solution situated on the acquired options in iterations.

Build Stage: An empty solution set used for initiating this stage. Every generation of this algorithm, y is chosen from the set N and all components from E. The components which might be suitable with the list N is placed on the list L and some of the optimal candidates are chosen from the list L and placed in optimal list OL.

Search stage: This stage is considered with change and eliminates functions. To generate the alternate solution for the component set observed in build stage, we use above stated two features. The change function makes the changes to the solution set through changing the component. The remove or eliminate function deletes the component from the solution set.

5. Experimental evaluation

We have considered the three development techniques for the component to validate the fuzzy logic engine. Along, with the development of component, the system considers the 5 exceptional contexts on the adaptation mechanism.

After attaining the adaptation, from the derived groups of solution, the group of accurate configurations of architecture practicable as regards the requirements of quality conferred. Therefore, the architect of software builds the compact opinion and selects best solutions. This work used the PSO for the CBSA. The JAVA is used for implementation of the work and the results are shown in the table 1. This proposed method used the source software as the Hospital Management System.

Number of components in a set is taken as 5, 10, 15, 20, and 25. After entire execution of an algorithm, the number of inconstantly provided interfaces and required interfaces are generated.

The GRASP is verified with a number of components, the time to reach the solution are mentioned in Table 1, and the optimum solution O founded by the approach is given. The PSO algorithm is tested with a number of components utilized by the GRASP approach. The time taken to gain the solution is much less when compared to Oavg is the optimal solution found after every 5 executions by applying metaheuristics. The time taken to achieve the solution is less when compared to the GRASP which is representing in figure 3. The LINGO optimized model is used to bought the outcome of the GRASP approach [19].

Table 1:PSO Algorithm Evaluation

Number of Components	GRASP		PSO	
	Time (s)	Optimal solution (O)	Time (s)	Optimal solution(OAVG)
5	43	76	8	77
10	45	42	11	47
15	47	63	17	71
20	49	52	20	55
25	50	65	26	69

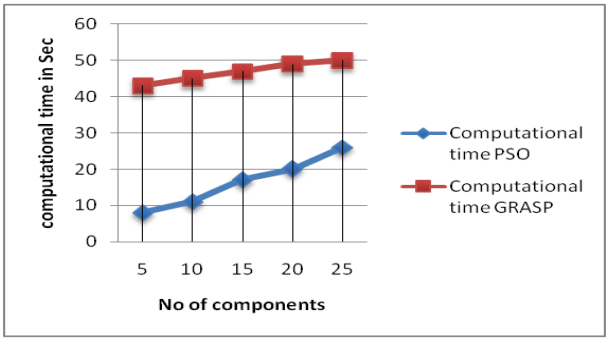

Fig. 2: Comparison of PSO and GRASP in Terms of Execution Time.

The efficiency of the PSO in terms of optimal solution is given in figure 3.To implement the software in the hospital management system; the system utilizes the OWL representation utilizing the Jena framework [20]. The fuzzy interface engine is implemented utilising MAT lab mathematical instrument [21].

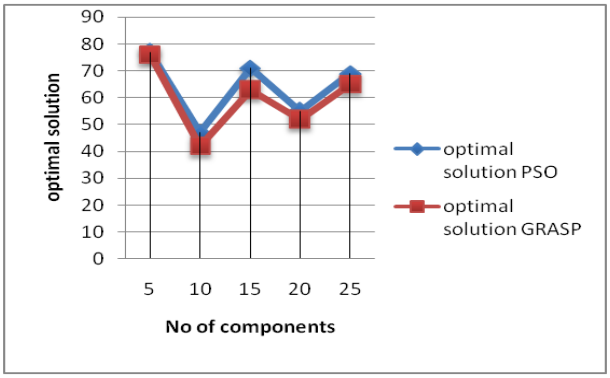

Fig. 3: Comparison of PSO and GRASP in Terms of OAVG.

6. Conclusion

This paper discusses a new strategy for component evaluation for component based adaptive software architecture. We outline the context aware policies to execute the component based on the functionality. The method follows the fuzzy logic to decide which components are adequate for the given context. The proposed an optimized approaches to find an acceptable component utilizing PSO algorithm. To attain the accurate results, the model is implemented using a case study by developing software for the Hospital Management. The outcomes acquired for the implementation is authentic, and the implementation of reconfiguration procedures are satisfactory.

References

[1] Gelernter, N. Carriero," Coordination Languages and their Significance", Commun. ACM 35 1992, 96.https://doi.org/10.1145/129630.376083.

[2] Zhang, Ji, and Betty HC Cheng. "Model-based development of dynamically adaptive software." Proceedings of the 28th international conference on Software engineering. ACM, 2006.

[3] De Lemos, Rogério, et al. "Software engineering for self-adaptive systems: A second research roadmap." Software Engineering for Self-Adaptive Systems II. Springer Berlin Heidelberg, 2013, 1-32.https://doi.org/10.1007/978-3-642-35813-5.

[4] Highsmith, Jim. "Adaptive software development: a collaborative approach to managing complex systems". Addison-Wesley, 2013.

[5] Seinturier, Lionel, et al. "A component-based middleware platform for reconfigurable service-oriented architectures." Software: Practice and Experience 42(5), 2012, 559-583.https://doi.org/10.1002/spe.1077.

[6] Y. MohanaRoopa, Dr. A. Rama Mohan Reddy, "Particle swarm optimization approach for component based software architecture", International Journal of Advanced Research in Computer Science and Software Engineering, 3(12),2013,557-561.

[7] Y. MohanaRoopa, Dr. A. Rama Mohan Reddy, "Component evaluation for Adaptive component based Software Architecture using Fuzzy Logic" Annals. Computer Science Series. 14th Tome 1st Fasc. – 2016, pp 18-24.

[8] O. Derin and A. Ferrante, "Enabling self-adaptivity in component based streaming applications,"SIGBED Review Special Issue on the 2nd International Workshop on Adaptive and Reconfigurable Embedded Systems (APRES '09), 6, 2009,1-35.

[9] P. Charles David and T. Ledoux, "Towards a framework for self-adaptive component-based applications," Distributed Applications and Interoperable Systems, 2893 of Lecture Notes in Computer Science, Springer, 2003, 1–14.

[10] K. Geihs, P. Barone, F. Eliassen et al., "A comprehensive solution for application-level adaptation," Software – Practice and Experience, 39(4), 2009, 385–422.https://doi.org/10.1002/spe.900.

[11] O. Derin, A. Ferrante, and A. V. Taddeo, "Coordinated management of hardware and software self-adaptivity," Journal of Systems Architecture, 55(3), 2009, 170–179.https://doi.org/10.1016/j.sysarc.2008.07.002.

[12] J. Combaz, J. C. Fernandez, T. Lepley, and J. Sifakis, "Fine grain QoS control for multimedia application software," in Proceedings of the Design, Automation and Test in Europe (DATE '05), 2,2005, 1038–1043.https://doi.org/10.1109/DATE.2005.155.

[13] J. Combaz, J. C. Fernandez, J. Sifakis, and L. Strus, "Using speed diagrams for symbolic quality management," in Proceedings of the 21st International Parallel and Distributed Processing Symposium (IPDPS '07), 2007, 1–8.https://doi.org/10.1109/IPDPS.2007.370341.

[14] M. Jaber, J. Combaz, L. Stras, and J. C. Fernandez, "Using neural networks for quality management," in Proceedings of the 13th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA '08), 2008, 1441–1448.https://doi.org/10.1109/ETFA.2008.4638586.

[15] E. Cannella, O. Derin, P. Meloni, G. Tuveri, and T. Stefanov, "Adaptivity support for mpsocs based on process migration in polyhedral process networks," VLSI Design, 2012, 17.https://doi.org/10.1155/2012/987209.

[16] Weyns, Danny, et al. "A survey of formal methods in self-adaptive systems." Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering. ACM, 2012, 67-79.https://doi.org/10.1145/2347583.2347592.

[17] G. Klir and B. Yuan," Fuzzy Sets and Fuzzy Logic", Prentice Hall, Englewood Cliffs, NJ, USA, 1995.

[18] M. Resende, "Greedy randomized adaptive search procedures (GRASP)," Encyclopedia of Optimization,Kluwer Academic Publisher, 2001, 373– 382.https://doi.org/10.1007/0-306-48332-7_188.

[19] M. Ramesh Babu, Y. MohanaRoopa," Component-based Self-adaptive Middleware Architecture for Networked Embedded Systems", International Journal of Applied Engineering Research, 12(12), 2017, 3029-34.

[20] The Apache Software Foundation, "Apache Jena," http:// incubator.apache. org/jena/. 2012.

[21] MathWorks, "Matlab—The Language of Technical Computing," http://www. mathworks.com/products/matlab/.2011.

[22] Hsu, Chao-Jung, and Chin-Yu Huang. "An adaptive reliability analysis using path testing for complex component-based software systems." Reliability, IEEE Transactions on 60(1), 2011, 158-170.https://doi.org/10.1109/TR.2011.2104490.

[23] Khakpour, Narges, et al. "Formal modeling of evolving self-adaptive systems." Science of Computer Programming 78(1), 2012, 3-26.https://doi.org/10.1016/j.scico.2011.09.004.